

APPLICATION
FOR
UNITED STATES LETTERS PATENT

TITLE: DISTRIBUTED SYNCHRONIZATION OF DATABASES
APPLICANT: DAVID J. BOOTHBY and ROBERT C. DALEY

"EXPRESS MAIL" Mailing Label Number EM51638015005

Date of Deposit Sept. 11, 1997

I hereby certify under 37 CFR 1.10 that this correspondence is being deposited with the United States Postal Service as "Express Mail Post Office To Addressee" with sufficient postage on the date indicated above and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

Lisa G Gray
Lisa G Gray

DISTRIBUTED SYNCHRONIZATION OF DATABASES

Reference to Microfiche Appendix

5 An appendix (appearing now in paper format to be replaced later in microfiche format) forms part of this application. The appendix, which includes a source code listing relating to an embodiment of the invention, includes
10 _____ frames on _____ microfiche.

This patent document (including the microfiche appendix) contains material that is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document as
15 it appears in the Patent and Trademark Office file or records, but otherwise reserves all copyright rights whatsoever.

Background

This invention relates to synchronizing databases.

20 Databases are collections of data entries which are organized, stored, and manipulated in a manner specified by applications known as database managers (hereinafter also referred to as "Applications"; the term "database" will also refer to the combination of a database manager and a
25 database proper). The manner in which database entries are organized in a database is known as its data structure.

There are generally two types of database managers. First are general purpose database managers in which the user determines (usually at the outset, but subject to
30 future revisions) what the data structure is. These Applications often have their own programming language and provide great flexibility to the user. Second are special purpose database managers that are specifically designed to create and manage a database having a preset data structure.

Examples of these special purpose database managers are various scheduling, diary, and contact manager Applications for desktop and handheld computers. Database managers organize the information in a database into records, with each record made up of fields. Fields and records of a database may have many different characteristics depending on the database manager's purpose and utility.

Databases can be said to be incompatible with one another when the data structure of one is not the same as the data structure of another, even though some of the content of the records is substantially the same. For example, one database may store names and addresses in the following fields: FIRST_NAME, LAST_NAME, and ADDRESS. Another database may, however, store the same information with the following structure: NAME, STREET_NO., STREET_NAME, CITY_STATE, and ZIP. Although the content of the records is intended to contain the same kind of information, the organization of that information is completely different.

Often users of incompatible databases want to be able to synchronize them with one another. For example, in the context of scheduling and contact manager Applications, a person might use one Application on the desktop computer at work while another on his handheld computer or his laptop computer while away from work. It is desirable for many of these users to be able to synchronize the entries on one with entries on another. The U.S. patent and copending patent application of the assignee hereof, Puma Technology, Inc. of St. Jose, California (U.S. Patent No. 5,392,390 (hereinafter, "the '390 patent"); U.S. Application, Serial No. 08/371,194, filed on January 11, 1995, incorporated by reference herein) show two methods for synchronizing

incompatible databases and solving some of the problems arising from incompatibility of databases.

Synchronization of two incompatible databases often requires comparison of their records so that they can be matched up prior to synchronization. This may require transferring records in one database from one computer to another. However, if the data transfer link between the two computers is slow, as for example is the case with current infrared ports, telephone modem, or small handheld computers, such a transfer increases the required time for synchronization by many folds.

Summary

In one aspect, the invention features a computer implemented method for synchronizing a first database located on a first computer and a second database located on a second computer. At the first computer, it is determined whether a record of the first database has been changed or added since a previous synchronization, using a first history file located on the first computer comprising records representative of records of the first database at the completion of the previous synchronization. If the record of the first database has not been changed or added since the previous synchronization, the first computer sends the second computer information which the second computer uses to identify the record of the first database to be unchanged.

The embodiments of this aspect of the invention may include one or more of the following features.

A second history file may be located on the second computer. The second history file contains records representative of records of the first database at the completion of the previous synchronization, where one of the

representative records represents the record of the first database determined to be unchanged. Then, at the second computer, a synchronization of the second and first databases is performed using the one of the representative records.

The information sent from the first computer to the second computer can be used to locate the one of the representative records in the second history file. The second history file can store information in relation to the representative records and the one of the representative records in the second history file can be identified from that stored information. Additionally, the information sent from the first computer to the second computer can include information that matches the information stored in relation to the one of the representative records in the second history files.

The information sent to the second computer can include information identifying records other than the unchanged record. It can also include information identifying the changed record. It can also include information identifying the deleted records or added records. The information can also include a code based on at least a portion of the content of the record of the first database. The code may be a hash number. The information may be a code uniquely identifying the record of the first database. Such a code may be one assigned by the first database to the records.

In another aspect, the invention features a computer implemented method of identifying a record of a database. A record of the database is read. A code is assigned to the record of the database, the code being based on at least a portion of the content of the record of the first database. The code is then to identify the record at a later time.

The embodiments of this aspect of the invention may include one or more of the following features.

The code may be a hash number computed based on at least a portion of the content of a record of the first
5 database.

The database is stored on a first computer and the code is transmitted to a second computer to identify the record to an application.

Advantages of the invention may include one or more
10 of the following advantages.

When synchronization is performed using the invention, a data transfer link, specially a slow data transfer link, is used efficiently, since unchanged records that are typically the majority of the records in a database
15 are not transferred between the two computers. Hence, when synchronizing two databases on two different computers, the time needed to synchronize the two databases is decreased

Also, when transmitting data from one computer to another, using a content based code, that requires less
20 bandwidth for being transmitted and nonetheless identifies a record, results in a slow data transfer links being used more efficiently.

The invention may be implemented in hardware or software, or a combination of both. Preferably, the
25 technique is implemented in computer programs executing on programmable computers that each include a processor, a storage medium readable by the processor (including volatile and non-volatile memory and/or storage elements), at least one input device, and at least one output device. Program
30 code is applied to data entered using the input device to perform the functions described above and to generate output information. The output information is applied to one or more output devices.

Each program is preferably implemented in a high level procedural or object oriented programming language to communicate with a computer system. However, the programs can be implemented in assembly or machine language, if
5 desired. In any case, the language may be a compiled or interpreted language.

Each such computer program is preferably stored on a storage medium or device (e.g., ROM or magnetic diskette) that is readable by a general or special purpose
10 programmable computer for configuring and operating the computer when the storage medium or device is read by the computer to perform the procedures described in this document. The system may also be considered to be implemented as a computer-readable storage medium,
15 configured with a computer program, where the storage medium so configured causes a computer to operate in a specific and predefined manner.

Other features and advantages of the invention will become apparent from the following description of various
20 embodiments, including the drawings, and from the claims.

Brief Description of the Drawing

Figure 1 shows two computers connected via data transfer link.

Figure 2 is a schematic drawing of the various
25 modules constituting an embodiment.

Figure 3 is a representation of the host workspace data array.

Figure 4 is pseudocode for the Translation Engine Control Module.

Figure 5 is pseudocode for a remote segment of a
30 synchronization program when loading records from and

unloading records to the remote database, when the database assigns unique IDs.

Figure 6 is pseudocode for a host segment of a synchronization program when loading records from and unloading records to the remote database, when the database assigns unique IDs.

Figure 7 is pseudocode for a remote segment of a synchronization program when loading records from and unloading records to the remote database, when the database does not assign unique IDs.

Figure 8 is pseudocode for a host segment of a synchronization program when loading records from and unloading records to the remote database, when the database assigns unique Ids.

15

Description

Briefly, referring to Figs. 1 and 2, a synchronization program, according to the embodiments described here, has a host segment 28 and a remote segment 26 which run on a host computer 20 and a remote computer 22, respectively. The two computer are connected together via a data transfer link 24 enabling them to transfer data between them. Data transfer link 24 may be a slow data transfer link such as a serial infrared links, serial cables, modems and telephone lines, or other such data transfer links. A host database 13 and a remote database 14, e.g. scheduling databases, are stored on remote computer 22 and host computer 20, respectively.

Generally, in some instances, both computers on which the two databases run are capable of running programs other than a database, as in the case of, for example, general purpose computers such as desktop and notebook

computers, or handheld computers having sufficient memory and processing power. In such a case, the synchronization program may be distributed between the two computers so as to, for example, increase the efficiency of using of a slow data transfer link between the two machines.

Briefly, at remote computer 22, remote segment 26 of the synchronization program loads records of remote database 13. Remote segment 26 then determines which records of the remote database have been changed/added, deleted or left unchanged since a previous synchronization. If the remote database assigns unique identification codes (i.e. unique ID) to its records, remote segment 26 can further differentiate between records than have been added and those than have been changed since the previous synchronization. Remote segment 26 uses a remote history file 30 which stores data representing or reflecting the records of the database at the completion of the previous synchronization. This data may be a copy of remote database 13. It may also be hash numbers for each of the records of the remote database. If the remote database assigns unique IDs, the remote history file may contain those unique IDs together with the hash numbers of the records corresponding to the stored unique IDs.

Remote segment 26 sends those records of the remote database that have been changed or added to the host segment or the host computer. However, the remote segment does not send the unchanged or deleted records to the host computer. Instead, the remote segment sends a flag indicating the status of the record (e.g. unchanged or changed) and some data or information that uniquely identifies the record to the host segment. This data or information may be a hash number of all or selected fields in the record at the completion of the last synchronization. It may also be the

unique ID assigned to the record by the remote database, if the database assigns one to its records.

Host segment 28 uses the received information or data that uniquely identifies the unchanged record to access a record in host history file 19 that corresponds to the received information or data. This record contains a copy of the data of the remote database record that the remote segment found to have been unchanged. Host segment 19 then uses this record to synchronize the databases by comparing it to the records of host database 14. After synchronization, the remote and host history files and the databases are updated. Since the unchanged records which typically constitute most of the records of a database are not transferred to the host computer, a data transfer link, specially a slow data transfer link, is used with increased efficiency.

We will describe two embodiments of a distributed synchronization program. We will first describe in general terms the overall structure of the distributed synchronization program in reference to Figs. 2 and 3 which is common to both embodiments. We will then describe then the first and second embodiments performing a distributed synchronization in reference to Figs. 4-8.

Fig. 2 shows the relationship between the various modules of an embodiment of a distributed synchronization program. Translation Engine 1 comprises a Control Module 2 that is responsible for controlling the synchronizing process by instructing various modules to perform specific tasks on the records of the two databases being synchronized. The Control Module 2 also provides data that affects the specific operation of the various components of the synchronization program, such as the name of the databases being synchronized and user preferences. Fig. 4

is the pseudocode of the steps taken by this module. The Synchronizer 15 has primary responsibility for carrying out the core synchronizing functions. It is a table-driven code which is capable of synchronizing various types of databases whose characteristics are provided by control module 2. The Synchronizer creates and uses a host workspace 16 (shown in detail in Fig. 3), which is a temporary data array used during the synchronization process.

A host translator 9 includes two modules: a reader module 10 which reads the data from the host database 14 and an unloader module 10 which analyzes and unloads records from the host workspace into the host database 14. Remote segment 26 also has similar modules for reading and unloading data from the remote database. The remote segment is designed specifically for interacting with remote database 13. The design of the remote segment is specifically based on the record and field structure of the remote database and remote database's Application Program Interface (API) requirements and limitations and other characteristics of the remote database. Similarly host translator 9 is designed specifically for the host database. The remote segment and host translator are not able to interact with any other databases or Applications. They are only aware of the characteristics of the databases for which they have been designed. In an alternate embodiment, the host translator and the remote segment can be designed as a table-driven code, where a general Translator is able to interact with a variety of databases based on the parameters supplied by, for example, the Control Module 2. It should be noted that the remote segment and host translator may be designed in various ways and still perform the tasks set out in this embodiment.

Fig. 4 is the pseudocode for the operation of Control Module 2 of the Translation Engine 1. We will use this pseudocode to generally describe distributed synchronization according to the invention. Control Module

5 2 first initializes itself and specifies the current user options to various modules (Step 401). In step 402, control module 2 instructs the Synchronizer to load host history file 19. Synchronizer 15 in response creates host workspace

10 16 data array and loads host history file 19 into host workspace 16. Host history file 19 is a file that was saved at the end of last synchronization and contains records representative of the records of the two databases at the end of the previous synchronization. Typically, the host

15 history file contains a copy of the results of the previous synchronization of the synchronized records of the two databases. It should be noted that the content of the records of the history file may be limited only to those fields that are synchronized and the data may be translated and stored in a format different than that of the remote

20 database or the host database. This data can be used to reconstruct the content of the records of the remote database as they were at the end of the previous synchronization. The host history file is generally used to determine changes to the databases since a previous

25 synchronization and also to recreate records not sent from the remote segment, as will be described in detail below. If no history file from a previous synchronization exists or the user chooses to synchronize without using the history file, in step 402 the synchronizer does not load a history

30 file. In that case, all the records from both databases will be loaded into the host workspace. We will describe the rest of the operation of the control module as if a history file exists and will be used.

Once the History File is loaded into the host workspace, Control Module 2 instructs host translator 13 to load the host database records (step 403). Host Reader module 11 of the host Translator reads the host database records and sends them to the Synchronizer for writing into the host workspace.

Control Module 2 then instructs remote segment to send the records of the remote database (step 404). Remote segment 26 reads the remote database records and sends them to Synchronizer 15 for writing into the host workspace. The actions taken by the synchronizer and the remote segment in response to step 404 will be described in detail in reference to Figs. 5, 6, 7, and 8, below.

Records in the host workspace are stored according to either the host database or the remote database data structures. Therefore, as synchronizer 15 receives each record, the Synchronizer maps that record using the appropriate record map (i.e. either a remote database to host database record map or a host database to remote database record map) before writing the record into the next available spot in the host workspace. Mapping may be performed by other modules, e.g. the remote segment. The records may also be "translated", i.e. cast into a format which synchronizer can use (a "translation" method is described in the '390 patent). For example, a date stored as "April 1, 97" may be translated into a format preferred by the synchronizer, e.g. "4-1-97".

Control module 2 then instructs the Synchronizer to perform a Conflict Analysis and Resolution ("CAAR") procedure on the records in the host workspace (step 405), which procedure is described in detail in the following applications of the assignee hereof, Puma Technology, Inc. of St. Jose, California, incorporated by reference in their

entirety including any appendices: "Synchronization of
Recurring Records in Incompatible Databases", Serial no.
08/752,490, filed on November 13, 1996 (hereinafter, "'490
application"); "Synchronization of Databases with Record
Sanitizing and Intelligent Comparison," Serial no.
08/749,926, filed November 13, 1996 (hereinafter, "'926
application"); "Synchronization of Databases with Date
Range," Serial no. 08/748,645, filed November 13, 1996
(hereinafter, "'645 application"). Generally,

synchronization is a process of analyzing records from the
remote database and host database against the records of the
history file to determine the changes, additions, and
deletions in each of the two databases since the previous
synchronization and what additions, deletions, or updates
need be made to the databases to synchronize the records of
the databases. Briefly, during CAAR, the synchronization
engine (i.e. the Synchronizer) compares the records in the
host workspace and determines what synchronizing actions
should be taken. The synchronization engine processes the
records, including comparing them to one another, in order
to form them into groups of related records. Each of these
groups may comprise at most one recurring or a group of
related nonrecurring records from each of the databases and
history file. After forming these groups from all records
of the two databases, the Synchronizer determines what
synchronization action should be taken. To do this, the
Synchronizer compares them, determines their differences,
and decides what synchronization action is appropriate or
asks the user what action should be taken. The synchronizer
then associates with that record, the specific "action"
(e.g. add, update or delete) that must be taken with respect
to that record in that record's database. During "CAAR",
the user may select not to synchronize a particular record

with the other database. We will describe below in detail the steps performed by the synchronizer and the remote segment in response to the output of CAAR as the output relates to the remote database.

5 Once Synchronizer 15 finishes performing CAAR on the records, the records may be unloaded or written into their respective databases, including any additions, updates, or deletions. However, prior to doing so, the user is asked to confirm proceeding with unloading (steps 108-109). Up to
10 this point, neither the databases nor the History File have been modified. The user may obtain through the Control Module's Graphical User Interface (GUI) various information regarding what will transpire upon unloading.

15 If the user chooses to proceed with synchronization and to unload, the records are then unloaded in order into the host database, the remote database and the History File. The Synchronizer in conjunction with the host translator and the remote segment perform the unloading for the databases. Synchronizer 15 creates a host history File and unloads the
20 records into it. Control Module 2 first instructs the host translator to unload the records from host workspace into the host database. Following unloading of the host records, Control Module 2 instructs the synchronizer and the remote segment to unload the remote records from the host workspace
25 (step 409). We will describe in detail below, in reference to Figs. 5-8, the specific actions taken by Synchronizer 15 and remote segment 26 in order to unload data from the host workspace into the remote database and the update remote history file 28. Control Module 2 next instructs the
30 Synchronizer to create a new History File (step 112). At this point Synchronization is complete.

Referring to Figs. 5 - 8, we will now describe the actions taken by the remote segment in coordination with the

Synchronizer in response to the instructions from control module 2 in step 404 to load records of the remote database and in step 409 to unload the records of the remote database from the host workspace. Specifically, we will describe two
5 embodiments. In the case of the first embodiment, the remote database assigns unique identification codes (i.e. unique IDs) to each of its records as they are created. In the case of the second embodiment, the remote database does not assign unique IDs to its records. Fig. 5 is the
10 pseudocode for the steps taken by the remote segment while Fig. 6 is the pseudocode for the steps taken by the Synchronizer in the case of the second embodiment. Similarly, Fig. 7 is the pseudocode for the steps taken by the remote segment while Fig. 8 is the pseudocode for the
15 steps taken by the Synchronizer in the case of the first embodiment.

Briefly, the remote segment determines which records have been changed/added, deleted or left unchanged since a previous synchronization. The remote segment uses a history
20 file located on the remote computer ("remote history file") to determine which records may have been changed/added, deleted or left unchanged since a previous synchronization. The remote segment essentially can translate outputs of any database into outputs of a fast synchronization database
25 which is a type of database that generally supplies information as to which of its records have been changed, added, deleted, or left unchanged. Fast synchronization databases and an example of a method of synchronizing them with other databases is described in detail in the '490,
30 '926 & '645 applications. Therefore, for example, this method of distributed synchronization may also be implemented with any synchronization program that is able to synchronize such databases.

Generally, the remote segment sends the host segment, over the data transfer link, only the content of those records that have been changed or newly added. As for unchanged records, the history file contains all necessary information to recreate or synchronize those records, if needed. Therefore, it is not necessary to transfer those records to the host segment. Only some data or identification code that uniquely identifies the record to the Synchronizer need be transferred for such a record.

Since the majority of records are typically unchanged records, not transferring them over the slow data transfer link improves the efficiency of the synchronization process.

After all necessary information has been transferred to the host segment, the Synchronizer synchronizes the databases. Following synchronization, the host segment transfers information necessary to update the remote database and the remote history file to the remote segment. The remote segment then updates its history file and the remote database.

Since both the host and remote segments rely heavily on history files to enable distributed synchronization, it is important that the host and remote segments use history files that correspond to one another, i.e. both contain records corresponding to a previous synchronization of the same two databases. In the described embodiment, the remote and host history files are named using a common naming convention. The name of a file is made up of six components:

- 1) Name or ID of the host computer, which may be an assigned name such as an assigned GUID in the case of operating systems by Microsoft Corporation of Redmond, Washington, or UUID in the case of operating systems by Open Software Foundation;

2) Name or ID of the host database application,
e.g. trademark designations "Lotus Organizer" or "Microsoft
Schedule+";

3) Name or ID of the host database file as stored
5 on the long term storage (e.g. hard disk drive) of the host
computer, e.g. "My Calendar";

4) Name or ID of the remote computer;

5) Name or ID of the remote database application;
and

10 6) Name or ID of the remote database.

Therefore, the remote segment and the host segment ensure
that the host history file have the same name. Moreover,
each of the history files have the date and time stamp of
the previous synchronization. The remote segment and
15 synchronizer use this to ensure that the history files from
the same previous synchronization of the two databases are
used.

Having described in general terms the actions taken
by the remote segment in coordination with the Synchronizer
20 in response to the instructions from control module 2 in
steps 404 and 409 (Fig. 4), we will now describe in detail a
first embodiment of their operation for the case where the
remote database assigns unique IDs to its records. We will
do so in reference to Figs. 5 and 6.

25 Fig. 5 is the pseudocode for steps taken by the
remote segment in response to the instruction by control
module in step 404 to load the remote database records into
the host workspace (Fig. 4). The remote segment first
initializes (i.e. creates) a remote workspace in the remote
30 computer (step 501). The remote segment then compares the
name of the host history file with the name of any remote
history file in the remote computer. If the remote segment
finds a remote history file that matches the host history

file (i.e. a remote history file that matches the host history file) (step 502), then the remote segment examine the date and time stamp of the host and remote history files. If the date and time stamp in the remote history
5 file matches the one in the host history file (step 503), then the remote segment determines that two history files correspond to one another. Hence, the remote segment loads the remote history file into the remote workspace.

In general, if matching history files do not exist
10 on the remote and host computers, the remote segment transfers all remote database records to the host computer. Therefore, if the name of the host and remote history files match but the date and time stamps do not match (step 505), then the remote segment assumes that remote history file is
15 not the correct remote history file to be used. The remote segment removes that history file (step 506) and transfers all remote database records to the host computer (step 507). If no remote history file matches the host history file (step 508), then the remote segment assumes an appropriate
20 remote history file does not exist. The remote segment transfers all the records to the host computer (step 509). To transfer all the records in the above steps, the remote segment first loads and stores all records of the remote database in the remote workspace. The remote segment then
25 transfers all records in the remote database to the host segment. If remote segment transfers all the records of the remote database to the host segment in either step 504 or 509, then the remote will go to step 528. It should be noted that the host segment will use the host history file,
30 if one exists, to perform the synchronization.

If an appropriate remote history file exists - i.e. conditions of steps 501 and 504 are satisfied - the remote history file is loaded into the work space. It is then used

to "filter" out information that need not be sent to the host segment since it already exists on the host segment. Generally, the history files on the remote and history files are used to store information representative of the remote database at the end of the previous synchronization. The records of the remote history file in the first embodiment contain the unique ID of the records and hash numbers of those records at the completion of a prior synchronization. In other embodiments, the remote history file may contain some or all of the field values of the records of the remote database.

Hashing may be described as converting any data, such as a string of characters, into a more compacted format, such as a number, meant to represent that string of characters. It may be considered to be a content-based encoding technique. The hashed values may be used as a surrogate for a hashed string of characters, for example, to compare strings. An example of a hashing algorithm is to calculate the following sum for every characters in a character string:

$$\text{sum} = \text{character} + (31 * \text{sum}),$$

where character is the number stored in the memory to represent that character (e.g. an Ascii value). (It should be noted that there are many ways of hashing data.) At the end of the computation, sum contains the hash number for that string of characters. In the described embodiments, the hash number is a 32 bit number and therefore can have a value between 2^{32} different values. Because the expected number of records is much less than this number, the probability of two different records having the same hash value is small. Therefore, hash numbers can be used to perform comparisons instead of comparing the non-hashed data or a preliminary check before comparing the data, with

relatively low risk inaccurate comparison. We have also use hash numbers as a unique identification code, which will be described in the second embodiment.

5 The remote segment uses the remote history file to determine whether a record has been changed, deleted, or added since a previous synchronization. Therefore, for records that are unchanged, which typically constitute the majority of records in a database, the remote segment sends information that the host segment can use to identify the
10 matching records in the host history file. That matching history file record contains the same data as necessary to use for synchronization as that on the remote database since the record is unchanged. Therefore, there is no need to send the whole record. In essence, the remote segment uses
15 the remote history file to filter out information that is already contained in the host history file and sending only those records that have been changed or added. In some embodiments, the remote history file may contain all the field values of the records of the remote database. In
20 those embodiments, the remote segment can determine not only which records have been changed but more specifically which field values have been changed. In that case, the remote segment can determine and then send only those field values that have been changed, further increasing the efficiency of
25 using the slow data transfer link.

We will now describe this process in detail. In the described embodiment, for each record of the remote database (step 515), the remote segment loads the field values, including the unique ID, of the record into the remote
30 workspace (step 512). As the records are loaded, they are translated (e.g. "translated" as described in the '390 patent) into a universal format for the remote workspace. The records will be translated back into the format of the

remote database as they are written into the remote database. The remote segment also computes a hash number based on all or selected (e.g. the fields to be synchronized) field values (step 513). In the described embodiment, the hashing number is a 32 bit number. The fields on which the hash number is based on remain the same for all synchronizations relying on this remote history file. The host segment also performs a hash on the same fields. If the fields which are hashed changes, the hash number of unchanged records would not remain the same from one synchronization to the next.

If the unique ID matches one of the unique IDs of records in the remote history file (step 515), then the record was present during the previous synchronization. That record could either be a changed record or an unchanged record. If the computed hash number for the record matches the hash number of the record in the history file (step 516), then the remote segment assumes that the record has not been changed since the previous synchronization and therefore can be created by the host segment from the host history file. The remote segment will take no action (step 517). In other embodiments, the remote segment can send the unique ID and a flag indicating that the record is unchanged to the host segment.

If the computed hash number does not match that of the history file record (step 518), the remote segment assumes that the record has been changed since a previous synchronization. Therefore, the remote segment sends the host computer the field values including the unique ID and a "changed" flag (step 519). In some embodiments, only those field values that have been changed since the previous synchronization will be sent, as described above. The remote segment then creates a new entry for the changed

record in the history file (step 520) and marks the record as unacknowledged (step 521), the purpose and function of which we will now briefly describe and is also described in the '490, '926 and '645 applications.

5 Generally, the remote segment does not change an entry in the remote history file, until it receives an instruction indicating that the host segment has synchronized and updated the host database with that record. This is done so that if for any reason (e.g. user does not
10 want to update that record of the host database as described above) the host database is not synchronized with that record, the remote segment will not treat that record as unchanged during the next synchronization. The acknowledgement may take the form of an "acknowledgment"
15 flag or an "action" instruction which instructs the remote segment to add, update, or delete that record of the remote database, as described above. Therefore, for each changed and deleted record, the remote segment creates a new entry and marks the entry as "unacknowledged". If an
20 "acknowledgment" flag is received, the old history file record is deleted. If an "acknowledgement" flag is not received, the new workspace entry is deleted. The steps will be described further below.

 If in step 515 the remote segment determines that
25 the unique ID of the loaded record does not match any of the unique IDs stored in the records of the history file (step 521), the remote segment assumes that the record loaded from the remote database has been newly added. Therefore, the remote segment sends the host segment a copy of the field
30 values of those fields of the record to be synchronized (which may be all or less than all the fields) together with an "added" flag (step 524). As in the case of a changed record, the remote segment creates a new remote workspace

entry and enters the unique ID and hash value of the record (step 525). The new entry is marked as unacknowledged (step 526).

After all the records have been loaded (step 528), the remote database determines that unique IDs in the history file that have not been matched represent the deleted records (step 529). Therefore, the remote segment sends the host segment those unique IDs together with "delete" flags (step 530).

After the remote segment has finished providing data to the host segment, the host segment synchronizes the two databases based on the input from the remote segment. The remote segment waits until the host segment finishes synchronizing and instructs the remote segment in step 409 in Fig. 4 to begin unloading into the remote database (step 532).

The host segment synchronizes the two database similar in the way it synchronizes a so-called "fast synchronization" database (as defined in the '490, '926, and '645 applications) with another database. The operation of a synchronization program synchronizing a fast synchronization database with either a fast synchronization database or a regular database (i.e. non- fast synchronization) is described in detail in the '490, '926, and '645. We will now describe in detail how the information from the remote segment is used to synchronize the remote database with another database.

As described above, a remote segment sending remote database records to the Synchronizer provides field values of only those records which have been changed or added since the previous synchronization but not those records that are unchanged or deleted. Therefore, unlike a regular database

Translator, the remote segment does not provide the Synchronizer with unchanged records.

In order to synchronize the remote database with the host database, the Synchronizer transforms information from the remote segment into regarding unchanged records into equivalent regular database records. These transformed records are then used by the Synchronizer in the synchronization. Essentially, the synchronizer transforms and uses the information sent by the remote segment to identify a record in the history file that is a copy of the field values of the unchanged remote database record. In the described embodiment, the synchronizer also copies that history file record and flags as being the remote database record.

The described embodiment uses the host history file to perform this transformation. At the beginning of a first synchronization between the two databases, all records in the remote database are loaded into the host history file. As changes, additions, and deletions are made to the remote database, during each subsequent synchronization, the same changes, additions, and deletions are made to the host history file. Therefore, the host history file at the end of each synchronization will contain a copy of the relevant content of the remote database after synchronization. By relevant, we mean data in the fields that are synchronized. For example, it may be the case that the host history file contain data in fields that are not synchronized. Moreover, if the records of the remote are mapped or recast into another format (e.g. "translated" as described in the '390 patent) the records of the history file contain a copy of the records of the database, as mapped, translated, or both. The Synchronizer uses the mapped or translated records for synchronization. Therefore, it only needs the mapped or

translated copy of the unchanged record. In other
embodiments, the host history file may contains copies of
all the records exactly as they are in the remote database
or in some other format that is useful for the particular
5 application.

Referring to Fig. 6, in the described embodiment,
all records received by the host segment from the remote
segment are flagged with one of Added, Changed, or Deleted
flags. For all records received from the remote segment
10 (step 601), the host synchronizer performs the following
functions. If a received record is flagged as an added .
record (step 602), then the received record is added to the
host workspace (step 603). Since the record is new, it is
not associated or linked to any history file record. If a
15 record is flagged as a "changed" record (step 604), then the
Synchronizer uses the received unique ID to find the
corresponding record in the history file (step 605) and
links the received remote record to that history file record
(step 606). If the received record is flagged as a
20 "deleted" record (step 607), then the Synchronizer uses the
received unique ID to find the corresponding record in the
history file (step 608) and marks the history file record as
deleted (step 609).

After all the received records are analyzed (step
25 611), if any host history file records containing remote
database unique IDs are left that were not matched against
the received records, the synchronizer assumes that those
records represent the remote database records that are
unchanged. For all those records (step 612), the
30 synchronizer clones the host history file record (i.e.
create a workspace entry and copy all the host history file
record in to that entry) and treats it as a record received
from the remote database. At this point the host segment

proceeds with synchronization since the records of the remote database have now been loaded. In essence, referring back to Fig. 4, this is the end of step 404.

As previously described, after the synchronizer has performed CAAR, the user must confirm to proceed with updating the remote database (step 406 in Fig. 4). If the user decides to terminate the synchronization, changes are not made to the host history file or the databases. In the case of the remote database, as described in reference to Fig. 5, the remote segment is waiting for the synchronizer to finish synchronizing. If the user aborts synchronization (step 533), the remote segment discards the remote workspace (step 534), saves the original history file without any changes (step 535), and terminates the process at the remote computer.

If the user confirms to proceed with updating the database (step 406 in Fig. 4), control module 2 instructs the synchronizer and the remote segment to proceed with unloading the records from the workspace into the remote database. As stated, at this point, the remote segment is waiting for the synchronizer to finish synchronizing (step 532 in Fig. 5). During the synchronization, the synchronizer has determined what "actions" with respect to which record in which database should be taken (update, delete, or add) to complete synchronization. If changes or additions are made to the host database in the case of particular record but no action need be taken with respect to that record in the remote database, the synchronizer determines that an "acknowledgement" should be sent to the remote segment. The synchronizer sends all the actions concerning the remote database together with the associated record to the remote (step 616). The synchronizer then sends the unique ID of those records that require

"acknowledgements" to be sent to the remote together with an appropriate flag (step 617).

Referring again to Fig. 5, for each action item or acknowledgement received at the remote segment (step 538), the following steps are performed. If the received data indicates an "acknowledgement" or "action" with respect to a record that was added or changed since the previous synchronization, the remote segment marks the new workspace entry that was created in either step 520 or step 525 as acknowledged (step 540). The remote segment also discards or removes any other entry in the workspace that contains the unique ID of this record, which is typically the entry that was loaded from the remote history file. Therefore, as previously described, this entry as opposed to the old remote history file entry associated with this record will be written into the history file at the end of the process at the remote segment. This in essence updates the history file, as will be described below.

If the received data indicates an action item that tells the remote segment to update, change, or add a remote database record (step 543), the remote segment performs that action with respect to the remote database. The remote segment also performs the same steps as steps 540 and 541 (step 544 and 545). If a new record was added to the database (step 546), it will be assigned a new unique ID. The remote segment sends that unique ID to the host segment (step 547). The host segment includes that unique ID in the host work space in association with that record (step 618 in Fig. 6).

After all the records have been received, the remote segment discards all unacknowledged entries from the workspace. Therefore, in the case of those added or changed records with which the user decided not to update the host

database, the remote history file remains unchanged. The remote history file is then updated from the remote workspace. At this point the control module continues with step 410 in Fig. 4, i.e. creating the history file to end the synchronization of the two databases.

In the first embodiment, which we described above, the remote database assigns unique IDs to its records. We will now describe a second embodiment for the case where the remote database does not assign unique IDs to its records.

In such a case, the remote segment provides some information less than all the fields of the records to uniquely identify an unchanged record to the host segment. This information may be a hash value. The host segment uses this information to find and then use the host history file copy of the unchanged remote database record to synchronize the two databases.

To identify a record from the previous synchronization or an unchanged record, the remote segment can use a content based code, that is a code whose value depends on the content of all or a selected number of the fields of a record. In the second embodiment, the remote segment uses hash numbers. Since in the case of an unchanged record, its content has remained the same, its hash number remains the same. The hash number acts as a unique identifier and therefore enables the remote and host segments to identify the unchanged record by its hash code. The hash code can be used to identify a record that is stored in the host history file, since its content remains the same from the end of one synchronization to the time it is updated. It may also be transmitted to identify an unchanged record or an unchanged version of a changed record. A host history file record can in effect be identified using the hash code of that record.

We will describe the operation of this embodiment in reference to Figs. 7 and 8. Steps 701 -711 are the same as steps 501-511 in Fig. 5, described above in reference to the first embodiment. These steps are generally concerned with finding the correct remote history file.

After determining that there is a suitable remote history file, for each record of the remote database (step 712), the following functions are performed. The remote segment loads and translates a record of the remote database into the remote workspace (step 713) and a hash number is calculated for that record (step 714). If the hash number of the remote record matches one or more hash numbers in the remote history file (step 715), then the remote segment assumes that the record has not been changed since a previous synchronization.

It is possible that the hash number may be repeated more than once, e.g. because of duplicate records or records that appear as duplicates because some of their fields are not synchronized. Therefore, the remote segment sends additional information that can be used to identify which of the multiple identical hash numbers a particular record relates to. This is done because during updating the remote history file record at the end of synchronization, the same number of identical hash numbers as matching remote database records are updated. In the second embodiment, this additional information is the index number associated with each entry of the remote workspace. Therefore, when the hash number of the remote record matches one or more hash numbers in the remote history file (step 715), the remote segment sends the hash number, a flag indicating that the record is unchanged, and the index number of that hash number to the host segment (step 716). Obviously if the

index number was previously sent, the next index number for the identical hash is sent.

If the hash number does not match one or more hash numbers in the history file (step 717), the remote segment treats that record as having been newly added. Therefore, the remote segment sends the host segment a copy of the field values of the record, the remote workspace index number, and an "added" flag (step 720). The remote workspace index number makes it easier to perform future search of the remote workspace when data with respect to this record is received. As in the case of changed and added record in the first embodiment, the remote segment also creates a new remote workspace entry and enters hash number value of the record (step 718). The new entry is marked as "unacknowledged" (step 719). It should be noted that although the remote segment treats the record as a new record, the remote segment can not distinguish between an added and a changed record. Therefore, the synchronizer during synchronization does not treat it as a new record. Instead, the synchronizer compares the record to determine whether it matches with any of host history file record which would mean it is a changed record.

After reading all the remote database records and processing them (step 722), the remote segment removes from the remote workspace all entries that have hash numbers that are unmatched (step 723). These entries represent records that have either been changed or deleted since the previous synchronization.

After the remote segment has finished providing data to the host segment, the host segment synchronizes the two databases based on the input from the remote segment. The remote segment waits until the host segment finishes synchronizing and instructs the remote segment in step 409

in Fig. 4 to begin unloading into the remote database (step 724).

Referring to Fig. 8, as in the case of the first embodiment, the synchronizer on the host computer uses the information to identify those records in the host history file that correspond to the unchanged remote database records. For every record received from the remote segment that is flagged as added (step 801), the synchronizer adds the record to the host workspace (step 802) and during CAAR compares the record to the history file to determine whether the record is a changed or added record. For every record received from the remote segment that is flagged as "unchanged" (step 804), in the same manner as the first embodiment, the synchronizer finds the corresponding host history file record by finding a record that has the same hash number as that sent by the remote synchronizer (step 805). The synchronizer then clones the record (step 806), as previously described, and treats as if it is a record received from the remote database. At the end of this process, when all the records of the remote database are loaded into the host workspace, the control module proceeds to step 405 in Fig. 4 to begin CAAR. CAAR will then analyze the records in the host workspace to determine which remote records were added, which were changed, and which were deleted since the previous synchronization.

After CAAR, if the user confirms to proceed with updating the database, control module 2 instructs the synchronizer and the remote segment to proceed with unloading the records from the workspace into the remote database (step 409 in Fig. 4). As stated, at this point, the remote segment is waiting for the synchronizer to finish synchronizing (step 724 in Fig. 7). During performing CAAR, the synchronizer has determined what actions should be taken

(update, delete, or add) to each database. If changes or additions are made to the host database in the case of a particular record but no action need be taken with respect to that record in the remote database, the synchronizer
5 determines that at least an "acknowledgement" is to be sent to the remote segment. The synchronizer sends all the actions concerning the remote database together with the associated record and remote workspace index to the remote (step 809). The synchronizer then sends the remote
10 workspace index of those records that require acknowledgements to be sent to the remote together with an appropriate flag (step 810). Therefore, the remote workspace index is used to identify which records in the remote workspace should be "acknowledged".

15 Referring back to Fig. 7, steps 725-729 are the same as steps 533-537, which were described in reference to the first embodiment. For each action item or acknowledgement received at the remote segment (step 730), the following steps are performed. If the data received indicates an
20 "acknowledgement" or "action" with respect to a record that was sent to the host segment flagged as "added" (step 731), the remote segment marks the new workspace entry that was created in either step 718 as acknowledged (step 732). It should be noted that the remote workspace index number is
25 used to locate the remote workspace entry. Therefore, as previously described, this entry will be written into the history file at the end of the process at the remote segment.

If the received data indicates an action item that
30 tells the remote segment to update, change, or add a remote database record (step 733), the remote segment performs that action with respect to the remote database. The remote

segment also updates the remote workspace and marks the entry as "acknowledge" (step 735).

5 After all the records have been received, the remote segment discards all unacknowledged entries from the workspace, which were newly created entries which were not acknowledged. Therefore, in case of those added or changed records with the user decided not to update the host database with, the remote history file remains unchanged. The remote history file is then updated from the workspace.
10 At this point the control module continues with step 410 in Fig. 4, i.e. creating the history file to end the synchronization of the two databases.

15 Although we have described embodiments in which the host segment transforms the input from the remote segment, it should be noted that other embodiments of the host segment may not transform the input from the remote segment since they are designed to use inputs that informs them of which records have been changed, added and deleted or have been left unchanged. Other embodiments in which the host
20 segment requires different types of input, the input from the remote segment are transformed as required. The various embodiments of the host segment may or may not use a history file.

Other embodiments are within the following claims.

25 What is claimed is: